徳島大学 工学部知能情報工学科 学部3年　居石 峻寛

Docker On CoreOS

ローカルにデプロイしよう

本スライドの目的

- Dockerを理解
- ついでにCoreOSも体験
- そのためにブレストアプリのプロトタイプをローカル環境にデプロイする

Gitは知ってますか？

いいところと不満

- Gitはソースコードのバージョン管理, 共有に非常に便利
- でもメンバ間の環境の差は埋められない
- ローカル環境 デプロイ環境間の差は埋められない
- 環境が異なると色々トラブルが起きる

Gitだけじゃ不便だよね！！

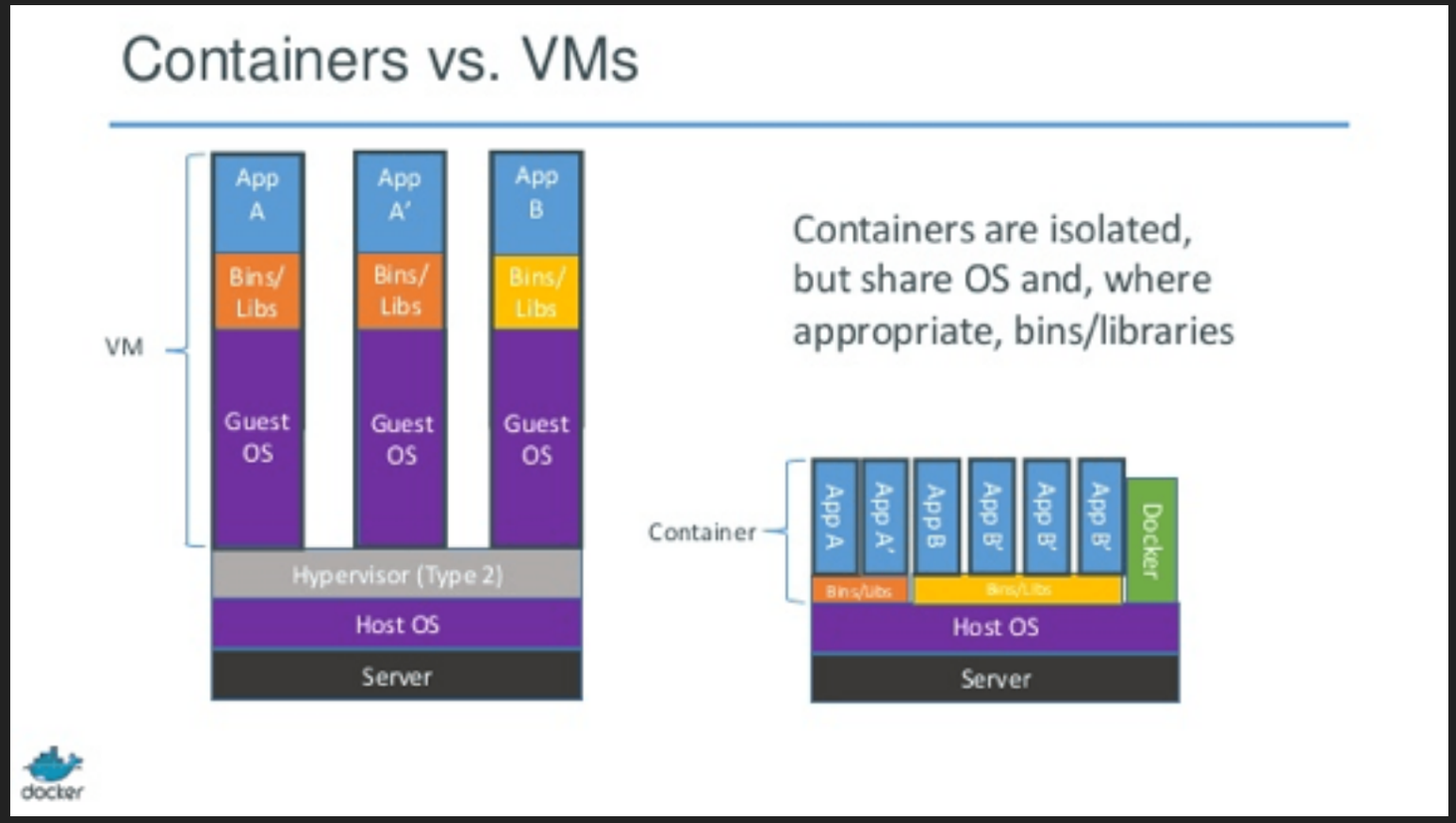
そこでDockerですよ

DOCKERってなに?

- 環境をcontainerという概念でパッケージングして持ち運ぶ
- Gitのように環境のバージョン管理も可能
- 同一マシン上で複数のcontainerをデプロイ可能

VMとの違い

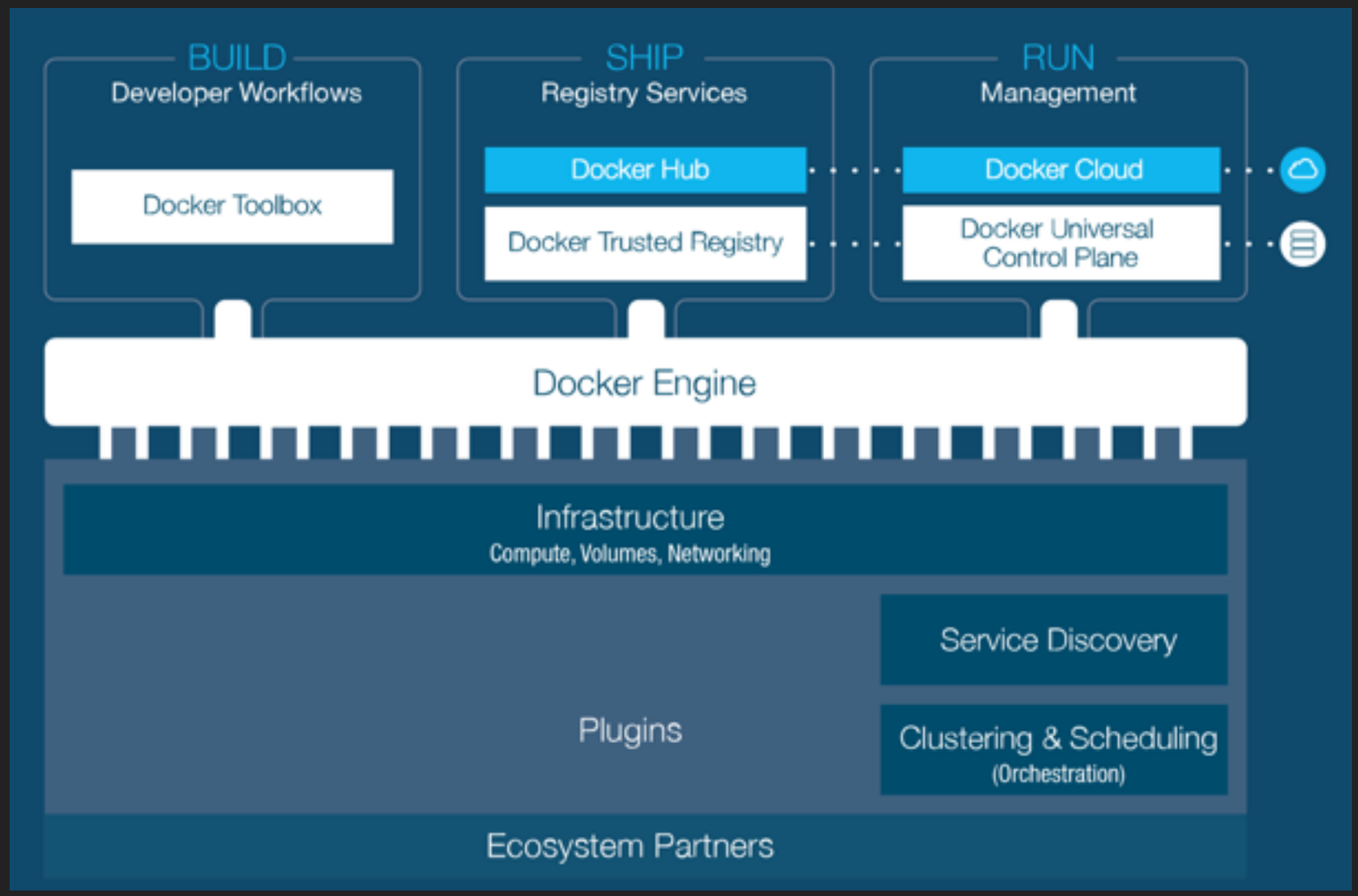
- 起動/処理が早い(Linuxカーネル等リソースをHostOSと共有)
- Appと環境はコンテナが持つ
- Dockerさえあれば動く
- ISOファイル不要



※ https://www.linkedin.com/pulse/containerization-docker-virtualization-vm-ahmed-banafa

BUILD, SHIP, RUN!

- Build (環境を構築)
- Ship (環境を運搬)
- Run (環境を運用)



※ https://www.docker.com/

Dockerはこの三つのステップを全面的にサポート！

(Runは有償のクラウドなのでローカルか自分のクラウドで我慢)

ではDOCKERの流れに沿って

- Build !
- Ship !
- Run !

Buildるには？

BUILDのパターン

- 自分で作る
- 他人が作った環境を引っ張ってくる
- 他人が作った環境を引っ張ってきてカスタマイズ

BUILDのパターン

- ~~自分で作る~~
- ~~他人が作った環境を引っ張ってくる~~
- 他人が作った環境を引っ張ってきてカスタマイズ

今回構築する環境

- ホストOSの上にVirtualBox
- VirtualBoxの上にCoreOS
- CoreOSの上にDocker
- Dockerの上にコンテナ
- コンテナの上に環境とApp

各環境の概要

Virtual Box + Vagrant

仮想的にOS環境を構築できる + 構築を手助け

CoreOS

超軽量OS, Dockerに必要な最低限のものだけ.

直接Appをビルドすることはほぼ不可能.

環境の作成と削除が楽

BOYS BE LAZY.

Takahiro Oriishi

早速やっていこ

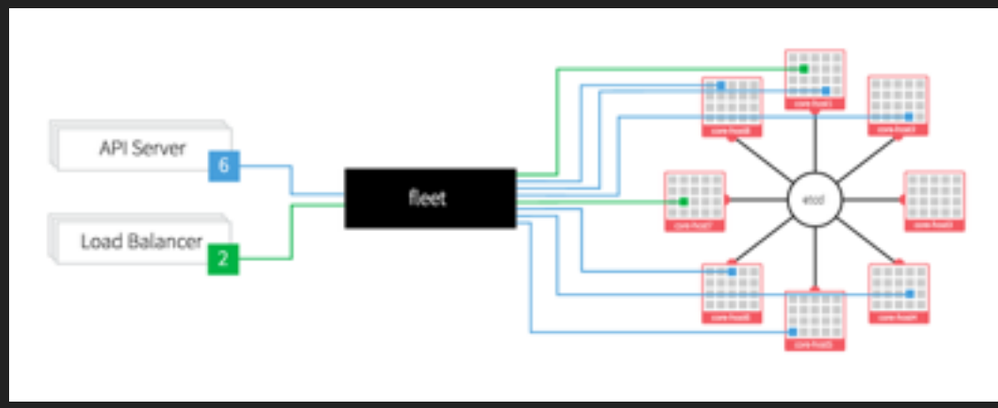
- git clone https://github.com/coreos/coreos-vagrant.git
- cd coreos-vagrant
- vagrant box list
- vagrant up (--provider={使用する仮想環境VirtualBox使うなら不要})
- vagrant status
- vagrant ssh

使用する仮想環境がVirtualBox以外ならプラグイン必要

(OSなんて)嘘じゃないですか
(ディレクトリの)
中に誰もいませんよ？

CORE OSはDOCKER運用の為のOS

- 最小限の機能+Docker搭載 (超軽量)
- Dockerにコア部分(Unixカーネル等)の提供をする踏み台
- etcd, rkt, fleet, flannel
  - etcd : クラスタで/etc(各種設定)を共有させる
  - rkt : Dockerコンテナでのクラスター管理を行う
  - fleet : クラスタの中で各マシンの状態に応じ自動スケジューリング
  - flannel : CoreOSクラスター内での内部通信を提供

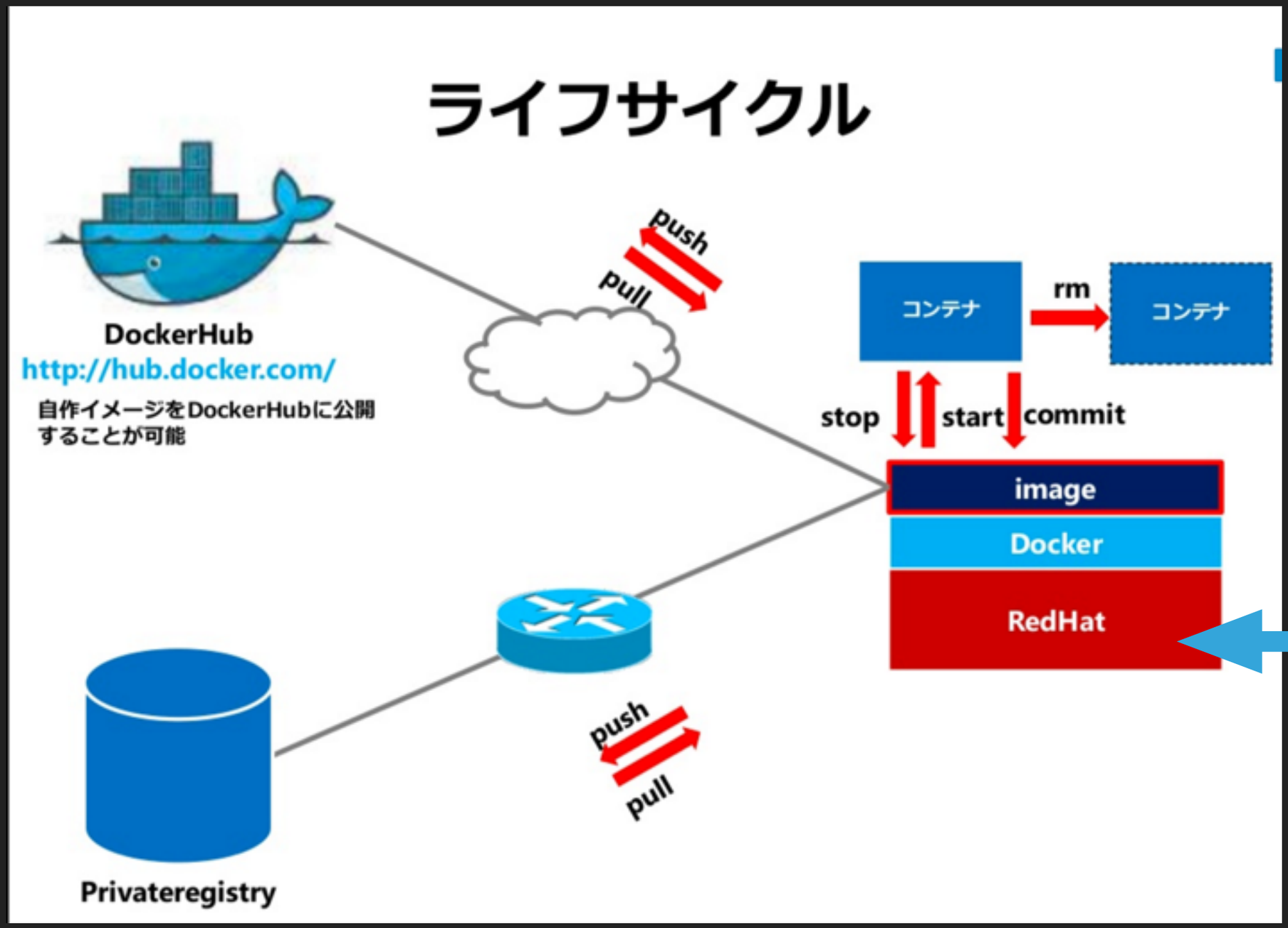


※https://coreos.com/using-coreos/clustering/

DOCKERいじいじ

- ifconfig (IPアドレスは要メモ!)
  - “172.17.8.”で始まる(ホストOSのcoreos-vangrant/Vagrantfileに書いてる)
- docker version
- docker pull oriishitakahiro/sample-img
- docker images

具体的なイメージ

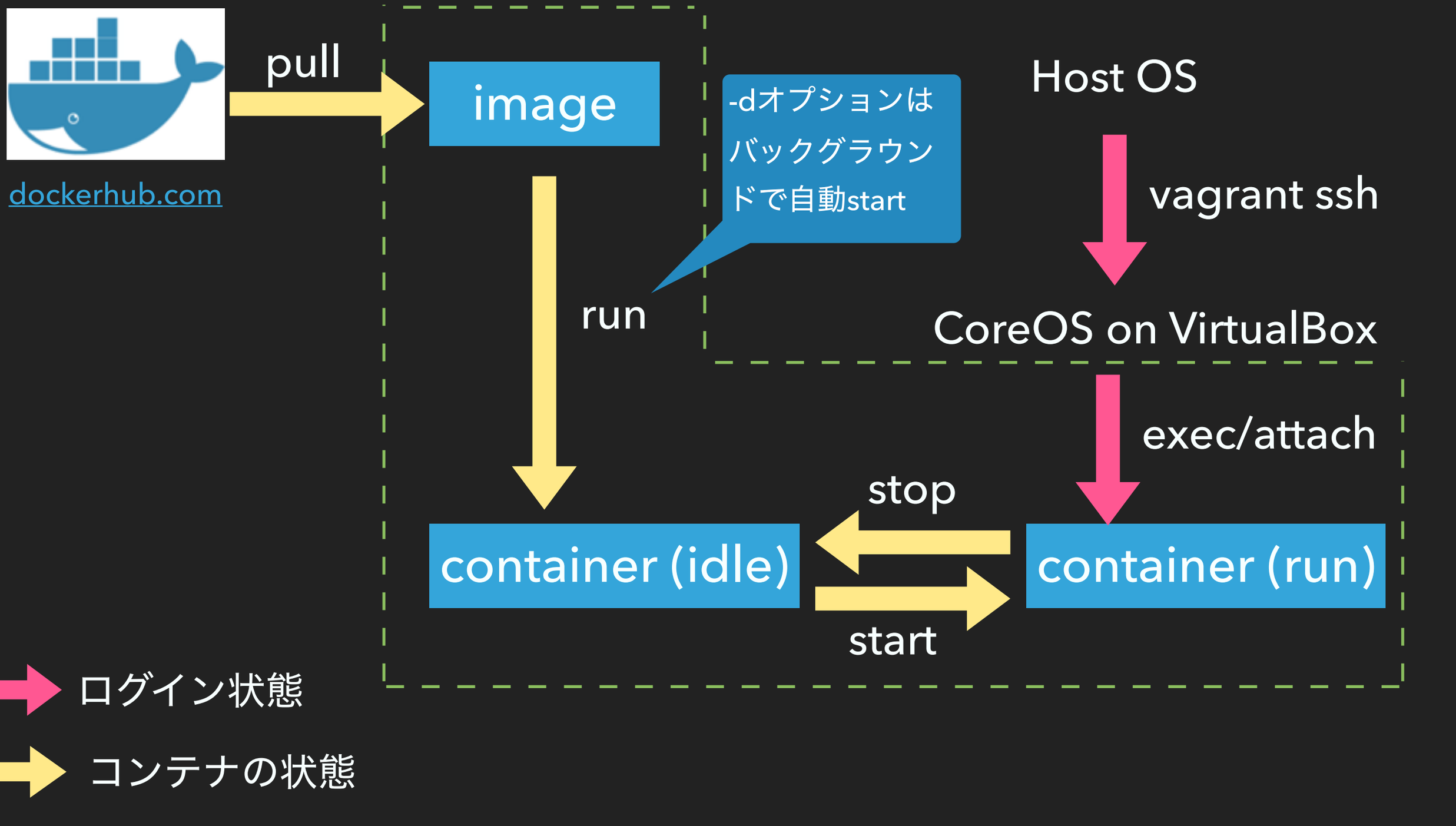


※http://www.slideshare.net/100003829577658/docker-54310989

DOCKERいじいじ

- docker run -d -it -p 80:80 --name samp <イメージID> /bin/bash
- docker ps (-a)
- docker exec -it <コンテナID> /bin/bash
  - docker attach <コンテナID>　※exitすると止まる

コマンドの流れ


pull
image
dockerhub.com
-dオプションは
バックグラウン
ドで自動start
run
Host OS
vagrant ssh
CoreOS on VirtualBox
exec/attach
stop
container (idle)
container (run)
start
ログイン状態
コンテナの状態

RUNコマンドの補足

docker run -d -it -p 80:80 --name samp <イメージID> /bin/bash

port forwarding

HostOS (CoreOS) の80番ポートとContainerの80番ポートを直結

“Host OS’s port No : Container’s port No”

複数のポートフォワードを行うことで、

同一サーバに複数の環境でWebAppを同時デプロイすることが可能

NGINXを動かしてみよう

- /etc/init.d/nginx start
  - さっき確認したipアドレスにブラウザからアクセス
- Welcome to nginxが表示されればOK!

RUBY周りの設定

- git clone https://github.com/OriishiTakahiro/brainstorming_app.git
- cd brain..
- rbenv local 2.2.0-dev
- rbenv rehash
- ruby -v
- rbenv exec gem install bundler
- rbenv rehash

RUBY周りの設定

- Gemfileに “ gem ‘therubyracer’ ”を追記
- rbenv exec bundle install

NGINXとRUBY ON RAILSの設定

- vim config/initializers/action_cable.rb
  - ActionCable.server.config.disable_request_forgery_protection = true
- vim config/application.rb
  - class Application の中に以下を追記
  - config.web_console.whitelisted_ips = %w(127.0.0.1 172.17.8.1 172.17.8.101)

NGINXとRUBY ON RAILSの設定

- vim /etc/nginx/conf.d/brainstorming_app.conf

```
upstream brainstorming_app {
        # Path to Puma SOCK file, as defined previously
        server unix:/var/run/brainstorming_app.sock fail_timeout=0;
}

server {
        listen 80;
        server_name 172.17.8.101;

        try_files $uri/index.html $uri @brainstorming_app;

        error_page 500 502 503 504 /500.html;
        client_max_body_size 4G;
        keepalive_timeout 10;

        include /etc/nginx/mime.types;
        root /var/app/brainstorming_app/public;

        error_log /var/log/nginx/error.log warn;

        # for web app
        location / {
                proxy_pass http://brainstorming_app;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header Host $http_host;
                proxy_redirect off;
        }
        # for action cable ( websocket )
        location /cable {
                proxy_pass http://brainstorming_app;
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "upgrade";
                proxy_set_header Host $host;
        }
}
```

NGINXとRUBY ON RAILSの設定

- vim config/puma.rb
- 以下をファイル先頭に追記

```
_proj_path = "#{File.expand_path("../..", __FILE__)}"
_proj_name = File.basename(_proj_path)

pidfile "/var/run/#{_proj_name}.pid"
bind "unix:///var/run/#{_proj_name}.sock"
directory _proj_path
```

RUBY ON RAILSをNGINXで動かしてみよう

- /etc/init.d/nginx checkconf
- /etc/init.d/nginx restart
- rbenv exec rails db:migrate RAILS_ENV=development
- puma -d
- ブラウザからIPアドレスへGO!

DOCKERHUBリポジトリにIMAGEファイルを上げてみよう

- docker commit -m “コメント” <コンテナID> <ユーザ名>/<リポジトリ名>
- docker images
- docker login - -username=<ユーザ名> - -email=<登録メールアドレス>
- docker push <ユーザ名>/<リポジトリ名>

pushはめっちゃ時間かかるので学校では止めような？